1 Introduction

As modern-day technology is ever evolving, the number of users who interact with technology increases consequently. As a result, the risk of an individual, or business, becoming a victim to cyber-crime increases proportionately. Small and medium-sized businesses (SMBs) are the biggest sectors targeted by cyber-criminals [1], which stem from issues such as budget restraints and expressing a lack of understanding towards cyber security concepts.

In fact, because of COVID-19 changing the dynamic of industry standards this year, a statistical analysis from May 2020 (UK) showed that individuals experiencing targeted hacking increased by 77.41% - in comparison with the previous year [2]. This is most likely since employees are encouraged to work from home via their personal computers. Consequently, this has fed into a new strategy whereby cyber-criminals are moving laterally into organisational infrastructure by targeting and infecting employees at their less secure personal computers [3].

Regarding this problem, this paper will explore the effectiveness of educational games - which has been shown to have an advantage on the learning outcome in comparison with traditional training material [4]. Therefore, this paper presents the following research question and hypothesis:

Problem Statement

Despite the existence of many cyber security awareness programs, there is still a lack of effective, widespread cyber security training

Research Question

Does teaching cyber security through a gamified medium improve user confidence in protecting against cyber-attacks?

Hypothesis

Creating an educational cyber security game will leave users feeling more confident and aware with regards to cyber security concepts

1.1 Goals

The goal of this project is to investigate how to effectively apply gamification mechanics to teach the OWASP # 10 cyber security weaknesses The expected result of this project is to create an accessible, online, multiplayer board game which could be used for both recreational and training purposes.

Application specific goals:

- The application will support 2-person multiplayer
- The application will be multi-platform (Desktop Application and Web-Accessible)
- The application will be playable cross-platform
- The application will illustrate a range of cyber security attacks and weaknesses
- The application will be extensible to support entirely new scenarios

1.2 Scope

As cyber-security is an incredibly broad field, only the OWASP #10 will be directly used to create an educational scenario of defending & attacking web applications. During the literature review, a range of pre-existing online game types are evaluated – with appropriate mechanics extracted and identified from Forde's research [5]. Using the findings from this evaluation, and the MDA framework [6], an appropriate serious game style is identified – with multiplayer being the core focus. Finally, the application is completed using Unity3D, Photon Unity Networking 2 and Photon Cloud.

2 Requirements

Based on the findings in the literature review, appropriate personas are constructed from the stakeholders who could benefit from the developed application. From this, requirements are identified and refined into user stories. These user stories are then prioritised according to the MoSCoW analysis and distributed into a Gantt Chart for a visual representation.

2.1 Stakeholder - Personas

From analysis of pre-existing cyber security games, a list of stakeholders in which educational cyber security games were developed and targeting includes:

- Students (Higher Education)
- Researchers and Professors
- Employees (SMBs)
- Security Consultants
- Security Advocates

Figure 1: Royalty - Free Image [7]

Persona One - Olivia - Student

Olivia is an undergraduate Physics student with a strong interest in programming and web development. Although she does not major in Computer Science, she would still like to learn more about the most common web application security risks.

Olivia would also love to educate her parents in basic cyber-security as her own father is not well acquainted with internet safety – and has succumbed to phishing emails himself.

Because her Dad suffers from colour blindness, her requirements is a colour-blind friendly game in which she could learn the rules and play with her father.

In this scenario, Olivia could benefit from the application by playing out multiple scenarios with her father – that would both inform her about secure web-development, as well as engaging her father with the world of cyber-security.

Figure 2: Royalty - Free Image [8]

Persona Two - Ben - Software Consultant

Ben is a software consultant for a medium-sized business. Although he has a degree in Computer Science, he is not an expert in cyber security.

As a consultant, Ben would like to develop software for his clients that is as secure as it is usable. Since this is done in a team, Ben would like to host an activity which brings together his co-workers (of varying experience) to participate in keeping up to date with current security trends.

His requirements are free, lightweight training material as he does not have the required funding, or time, to hire an expert security consultant.

To benefit his goal, the application will have multiplayer that will encourage his employees to challenge each-other in a stimulating way. Furthermore, the gamification aspect will not require his co-workers to work hard outside of their contracted hours.

Figure 3: Royalty - Free Image [9]

Persona Three - Samira - Security Analyst

Samira is a senior level security analyst. Samira completed her PhD on using gamification to teach security.

As an expert in her field, her passion for cyber security extends to all her colleagues, friends, and family. As such, she has a strong desire in raising awareness and eliminating the human error which lead to most security exploits.

Samira would benefit from the application by using it as a framework to map the mechanics to alternative cyber-risk scenarios, thus creating a training tool. Furthermore, she could use the application as a foundation to further her own research in gamification techniques.

2.2 Functional Requirements

The following requirements are derived from the findings in the literature review, as well as specific features that proposed application should have.

ID	Requirement	Description	MoSCoW
1	Multiplayer	Multiplayer turns between connected users	Must
2	Matchmaking	Users can host and join a game session	Must
3	Quit / Leave	Users can leave at any point	Must
4	Web Accessible	Playable on a web browser	Should
5	View Units	Users can see the status of all units	Should
6	View Description	Units and moves have an explanation	Must
7	View History	Users can see the the game sequence	Must
8	Complete	Users can play a full version of the game	Must
9	Clear Goal	Gameplay loop and objectives are clear	Must
10	Single Player	Users can play single player	Won't
11	Movement	Units move with varied distances correctly	Must
12	Unit behaviour	Units have unique stats, moves and abilities	Must
13	Tutorial	Users can learn the rules and mechanics easily	Must
14	Gamification	Gamification mechanics should encite learning	Must
15	Challenge	The game should invoke critical thinking	Must

Table 1: Functional Requirements

2.3 Non - Functional Requirements

The following requirements outline how the application should perform. Although these are not required to reach a minimum viable product, they benefit the effectivity of the application greatly.

ID	Requirement	Description	MoSCoW
16	Availability	Cross-platform playable (e.g. Tablet browser)	Could
17	Ease of use	Easy to learn, understand and use	Should
18	Accessibility	Compliant with WCAG 2.1 and colour-blind friendly	Should
19	Secure	Playable on a secure website domain	Should
20	Correct	Factually correct in any cyber security concepts explored	Must
21	Performance	Gameplay should be 30 fps with low multiplayer latency	Must
22	Scalable	Scalable to support more than 2 players at any given time	Should
23	Learning	Users can learn deeper cyber security principles	Must
24	Extensible	Flexibility to add different scenarios, units and abilities	Won't
25	Feedback	Provides custom feedback based on performance (e.g. Chess)	Could
26	Music	Good music that fits the theme and aesthetic	Should
27	Animations	Unit movement, and abilities, have satisfying animations	Should
28	Enjoyable	Depth of mechanics and dynamics that lead to meaningful play	Must

Table 2: Non-Functional Requirements

2.4 User Stories

ID	As a (role/user)	I want a/to.. (feature)	So that I can... (benefit)	MoSCoW	Ref	Difficulty
1	General Player	View a unit's stats (HP/Defence)	Know who to attack	Must	5,6	Low
2	General Player	View an abilitiy's range, damage & cost	Know what I can do	Must	5	Low
3	General Player	View a colour-blind friendly UI	Interact with the application easily	Should	17	Low
4	General Player	View a unit's movement range	Where I can move to	Must	11	Medium
5	General Player	View a unit's ability attack range	Know who I can reach	Must	12	Medium
6	General Player	Move a unit	Attack a target	Must	8,12	High
7	General Player	Attack a target	Kill an enemy unit	Must	8,13	Medium
8	General Player	Disable the target	Prevent them from moving	Must	8,14	Medium
9	General Player	Defend an ally unit	Prevent them from dying	Must	8,15	Medium
10	General Player	Click on a move description	Learn more about the attack nature	Must	6	Low
11	General Player	Click on a unit	Learn more about the attack vector	Must	7	Low
12	General Player	Click on and view a history log	Follow the game's sequence of events	Should	7	Low
13	General Player	Click on a unit map	Know how units relate to eachother	Could	26	High
14	General Player	View the units remaining	I know what units I can still move	Should	5	Low
15	General Player	Pan the camera and/or zoom in	See the map from different angles	Could	20	Medium
16	General Player	Unique animations for each ability	Differentiate abilities	Could	24,28	Very High
17	General Player	Varied unity abilities	Enjoy varied gameplay	Could	12,24	High
18	General Player	An intermediate tutorial	Quickly learn how to play	Should	13	Very High
19	General Player	A help / status bar	Understand the gameplay	Should	13,23	Medium
20	General Player	Multiplayer functionality	Host and join a session	Must	1,2,8	Very High
21	General Player	Singleplayer functionality	Play solo	Won't	10	N/A
22	General Player	Quit / leave the session	Play with a friend	Must	3,8	Low
23	General Player	Win or lose	Finish a game session	Must	8	Low
24	Security Analyst	Host a workshop session	Organise fun, training sessions	Could	1,2,14	High
25	Student	Learn about the OWASP #10	Understand attacks and mitigations	Must	14,23	Low
26	General Player	Purchase & upgrade new units	Have a deeper reason to keep playing	Could	12,25	Very High
27	Security Analyst	Receive feedback after a game	Know how to play better next time	Could	26	Very High
28	General Player	Hear gameplay music	Become immersed within the game	Should	24,27	Medium

Table 3: Identifying User Stories from the Personas and Application Requirements

2.5 Constraints and Limitations

- The application has to be optimised with simple textures to run on a web browser due to rendering limitations in WebGL
- Different web browsers have different rendering abilities
- The application will have resizing and mouse interactivity limitations on different devices
- Given the development time limit, only one game "scenario" will created.
- It will not be possible to iteratively develop a game after sufficient play-testing and receiving user-feedback
- Due to Covid19, it will not be possible to create a physical paper prototype of the application, and evaluate this beforehand

3 Project Management Tools & Techniques

The following tools and techniques were used for the development of this application and report. Is it this better place towards the end of the report (in evaluation), along with a table for techniques, e.g. agile methodology, weekly meetings, MoSCoW priority etc?

Tools	Description
GitLab	Project management (Boards, Issues and Milestones) & version control
GitHub Desktop	Local software for handling version control
Trello	Visual representation for setting daily and weekly tasks
Workona	Chrome extension for streamlining online research into a succinct workplace
Menderley	Reference management for supporting literature
Microsoft Teams	Communication software for weekly remote meetings
Google Drive	Cloud storage for research, recording minutes and sharing documents
Lucid Chart	Online software for UML diagrams
Unity3D	Game engine for developing for the Web (WebGL)
JetBrains Rider	Integrated development environment for C#, .NET & Unity
Photon Cloud	Cloud service for hosting multiplayer servers online
PUN2	Multiplayer networking API
ParralelSync	Open source software to clone the Unity development environment
Itch.IO	Secure domain to host WebGL and HTML games

Table 4: Project Management Tools & Techniques

3.1 Risk Assessment

In the risk assessment below, the Risk Exposure (R.E) is given by the product of the Probability and Severity

Risk	Prob (1-5)	Sev (1-10)	R.E	Mitigation
Project deadlines not met	3	10	30	Establish soft deadlines and meet with supervisor weekly for continuous feedback & self-evaluation
Online multiplayer failure (limitations/pricing of cloud server hosting)	3	9	27	Develop LAN multiplayer functionality from Unity Mirror API, or local multiplayer co-op
Not obtaining Ethics Approval in time	3	8	24	Refer to generic ERGO extension
Application not effective at teaching	3	7	21	By identifying an appropriate target demographic and the most appropriate cyber security content to teach
Over-estimating scope of implementation	3	7	21	Plan project into smaller iterations and prioritise according to MoSCoW
Application does not relate to original problem statement & hypothesis	2	9	18	Continuously referring back to the initial problem statement and hypothesis
Application does not meet user requirements	2	8	16	Prioritise musts, shoulds then coulds, and functional requirements before non-functional
Sickness, flu or mental health difficulties	2	8	16	Exercising daily, eating healthiy, breaking down tasks into small achievable chunks, and reach out for support if needed
Impact of Covid19	5	3	15	Application will utilise online multiplayer (for remote gameplay), alongside Microsoft Teams, Discord & GitLab for communication and development
Gamified mechanics not implemented appropriately	2	7	14	By identifying key mechanics through literature and game review, and use the MDA framework to decide upon the decided mechanics for the game
Stolen work/data from: cloud storage being compromised & ransomware	1	10	10	Using a secure password manager, 2FA and manually backing up important files on multiple locations
Loss of work/data	1	10	10	Git version control and OneDrive/GoogleDrive for cloud storage
Under-estimating scope of implementation	1	5	5	Further refine/iterate the game after evaluation, and implement all Coulds if time permits

Table 5: Risk Assessment

3.2 Gantt Chart

Haven't updated gantt charts yet - is placing this better situated later on (evaluation/conclusion)?

4 Design

This section gives a top-level overview of the physical design, and the theory behind each implementation of the project.

4.1 Applying the MDA Framework

The MDA (Mechanics, Dynamics, Aesthetics) framework [6] is the most well-known framework to identify the requirements of building a game. It is an iterative process in which the mechanics of the game are first outlined, and evaluated, to see what game-play loops (dynamics) emerge. Finally, the dynamics are mapped to a target aesthetic– e.g. the scarcity of resources is an important dynamic for a survival title.

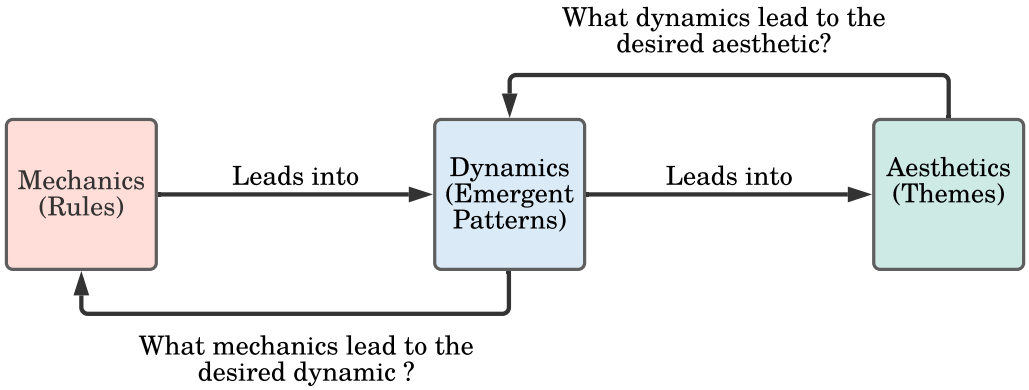


Figure 4: The Mechanics-Dynamics-Aesthetics Model

4.1.1 Mechanics

The mechanics are fundamentally the most basic rules and actions a player can do within the game world [6]. They can be identified as follows:

- Action Point (AP) requirement for certain games
- Moves limited by a distance range and can be blocked by other
- Movement opportunities is blocked by both enemy and friendly units
- Units can only attack once and move once per turn
- Units can do damage, bypass shields, restore defence and disable other units
- Disabling a database / web server unit will stop AP gain from that unit for one turn
- Disabling movable units will prevent from acting for one turn
- Units have a max defence threshold cannot be restored past the threshold

- When a unit has 0 HP, it will be removed from the game world
- Units cannot move after attacking (even if they did not move that turn)
- Each movable unit has 2 abilities, whereas the static units (database / web server) cannot directly be interacted with

4.1.2 Dynamics

The dynamics can be thought of as 'the emergent behaviour that arises from gameplay when the mechanics are implemented' [6].They can be identified as follows:

- Players may choose to target the web server/database first to reduce AP gain
- Players may use shield destroying moves on heavy units
- Players may use cheaper moves to save AP
- Players may block their database/web server to protect them
- Players may use the bypass defence move to target units with a lot of defence but little HP (e.g. web server)

4.1.3 Aesthetics

The aesthetics can be thought of as 'the emotional response a game should illicit from the player' [6]. They can be broken down into:

- Sensation (emotion-invoking)
- Fantasy (immersion)
- Narrative (story - rich)
- **Challenge (puzzles / obstacles)***
- **Fellowship (co-operative / multiplayer)***
- Discovery (open world)
- Expression (character creation)
- Submission (simulations)

* Where Fellowship and Challenge are the two key target Aesthetics for the purpose of this application.

4.2 Mapping the OWASP Top 10 to Game Mechanics

Figure 5 represents the choice of mapping security risks, exploits and mitigations to in-game mechanics. The nature of these mechanics are explored in section **??**. The OWASP Top 10 was chosen as it represents a broad consensus of the most common web application security risks [10].

With regards to figure 5:

- Green represents an example of attack derived from the OWASP Top 10
- Red represents an attack that exploits a security risk
- Blue represents a potential mitigation towards a security risk

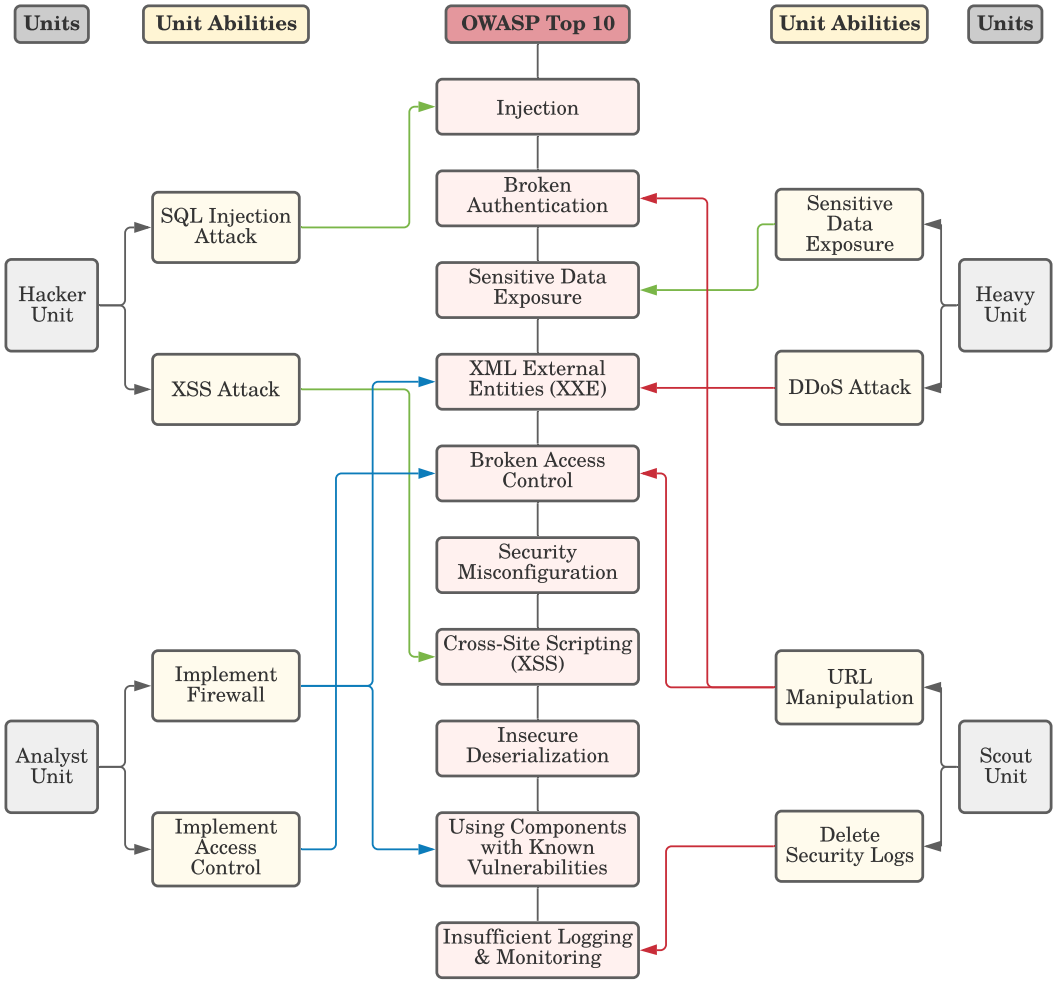


Figure 5: Mapping the OWASP Top 10 to Game - Mechanics

Need to summarise the content in this table and reference properly

Ability	Description	ID
SQL Injection	Sequel Query Language (SQL) injection exploits the interactivity between the web application and commands that query a backend database. Using SQL injection, it is possible to bypass authorisation, extract sensitive information from databases and even manipulate database entries with your own custom information.	1
XSS Attack	Cross-Site Scripting (XSS) attacks can occur when a web application fails to validate user input by escaping special characters. As a result, an attacker can execute scripts in a victim's browser, hijack their session data and even redirect them to other phishing sites.	7
Firewall	Firewalls block and filter unwanted traffic to keep the web application isolated within a safe environment. Furthermore, firewalls can be used in conjunction with an Intrusion Detection/Prevention system to detect malicious activity and send alerts upon detection. A firewall can also be used to mitigate XML External Entities attacks (XXE)	4,9
DDoS	A DDoS attack involves using a network of devices to flood a web application, network or server with overwhelming internet traffic - and disable it from being able to handle requests. Although not directly listed in the OWASP Top Ten, Denial of Service attacks can be accomplished from exploiting XML External Entities (XXE)	4
URL Manipulation	URL Manipulation changes the URL parameters to bypass authorisation in websites that suffer from broken access control. The goal is to access admin locations, root files and sensitive data (such as software versions) which could be used to identify vulnerabilities within the system.	2,5
Sensitive Data Exposure	Examples of sensitive data include personal, account and financial information. Improper handling of sensitive data in transit or storage lead to data breaches which can cripple a company's reputation. In more severe outcomes, such information could be used to commit fraud and identity theft.	3
Delete Security Logs	Insufficient data logging and monitoring allows an attacker to persist in a compromised system - undetected for a longer time without being detected. Logging pertains to recording anomalies, failed login attempts, user activity and so on, whilst effective monitoring is to know how to interpret the logged data - and how to react appropriately in the case of such detection.	10
Access Control Systems	Broken access control is present when employees/users have access to higher levels of clearance, sensitive information and possibly even "admin" operations than they require.A good access control system should be implemented once, and periodically reviewed. Control policies based on user roles should enforce the least amount of privileges each user requires to complete their interaction within the web application.	5

Table 6: A Description of Ability Names

4.3 UML Modelling - A Class Diagram Representation

Figure 6 represents the core basis of functional code within the application but omits all testing, utility and helper scripts.

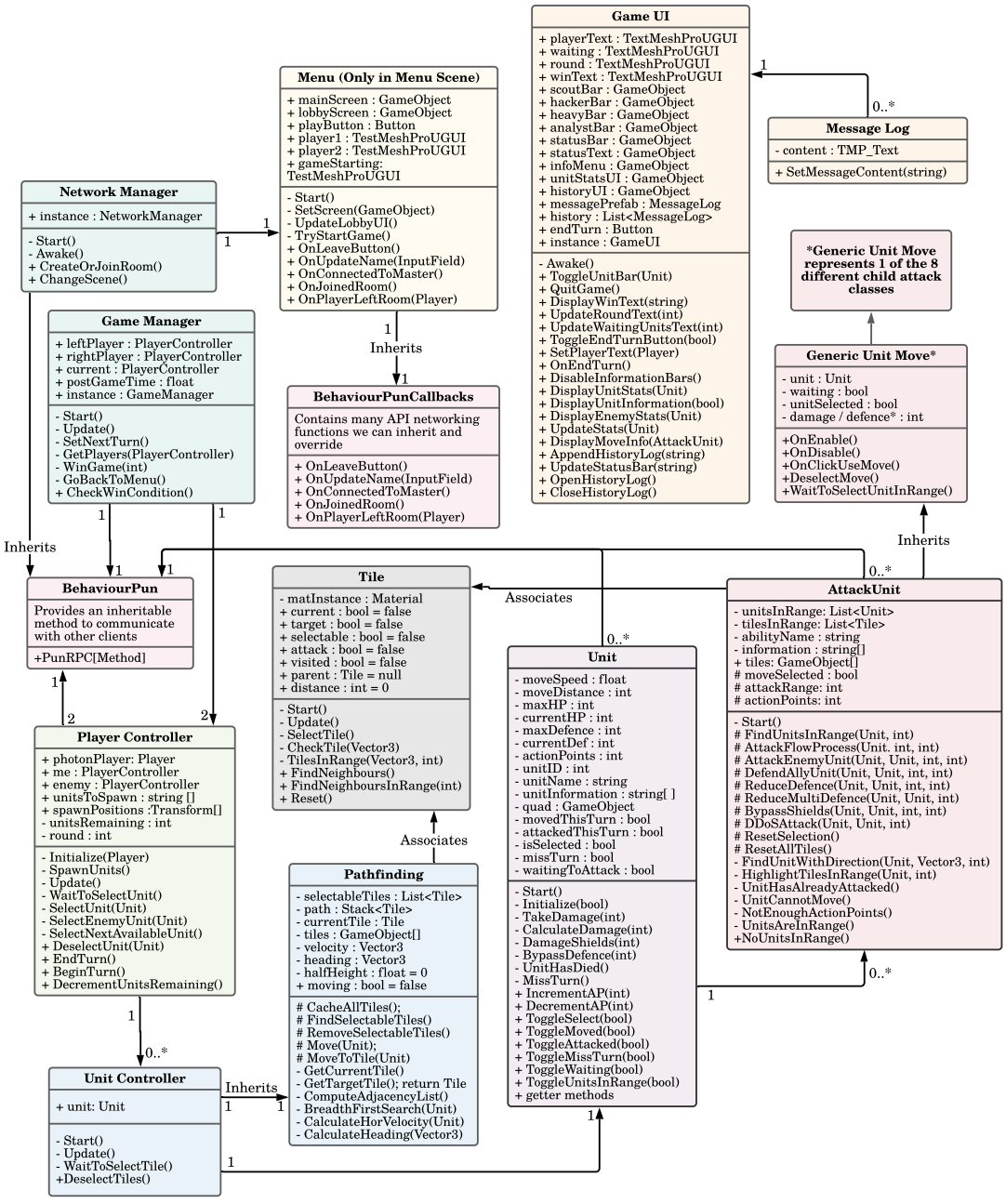


Figure 6: A Class Diagram Representation

4.4 Mapping the MDA Framework to Class Functions

Figure 7 highlights functions and components that specifically relate to the MDA framework. Due to the nature of Aesthetics being an emotional product of the game, in this diagram it represents all the physical assets and UI that the player interacts with.

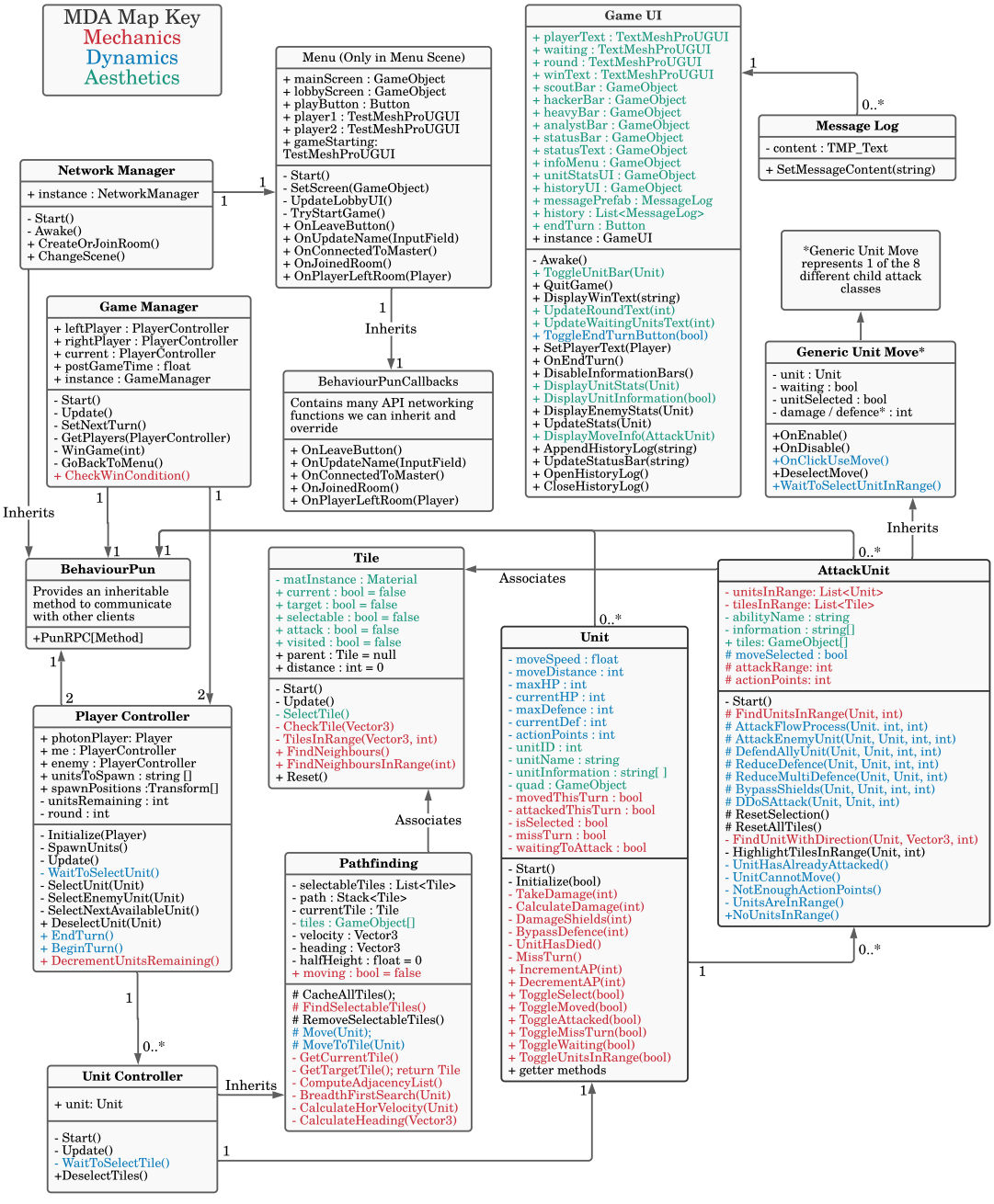


Figure 7: Mapping the MDA Framework to Class Functions

With reference to figure 6 and 7, an explanation of the core classes and components can be found below:

Game Manager
A singleton instance of this class forms the backbone of the project as it controls the game's flow sequence, turn-manager, and game scene.

Behaviour PUN Callbacks
This class is provided by the Photon Unity Networking 2 API and provides many networking functions that can be implemented and overridden.

Network Manager
A singleton instance of this class inherits the PUN2 API to handles hosting, joining, and leaving a room.

Menu
This is a relatively simple class that takes user input (from within the game menu) to invoke networking functions within the network manager.

Game UI
This script handles the transition and display of the user interface (UI) components.

Player Controller
A singleton instance of this class is instantiated for each player. It listens for player input once per frame. It allows the player to select units and manages the players-turn and status of all units each turn.

Unit Controller
Each Unit component has an instance of this class. It enables tile-selection and unit movement by inheriting functions provided from the Pathfinding class.

Path-finding
This class utilises Breadth First Search to find the range of tiles that a selected unit can move to. It directly associates to each tile as it checks if it's unoccupied, and then highlights it green if it is a valid destination. The pathfinding functions within this class were loosely adapted from a tutorial available at Game Programming Academy [11].

Unit
Each Unit component has an instance of this class. It contains unit information (e.g. health, defence, etc..) and functions relating to receiving damage, restoring defence and the unit's status (e.g. the unit has moved).

Attack Unit
Attack Unit provides functions for finding units in range, attacking, and defending units. It associates with Tile class to accomplish this (e.g. is a Unit located at this tile).

Generic Unit Move
Generic Unit Move represents the 8 unit abilities which are derived from the OWASP Top 10 (e.g. SQL Injection). These scripts inherit from Attack Unit for their shared functionality and are directly linked to the UI buttons which invoke the ability event upon user input.

4.5 UML Modelling - Activity Diagram

The following diagram refers to the sequence of two players joining a game, matchmaking and then the typical sequence of events that happens until the game is finished.

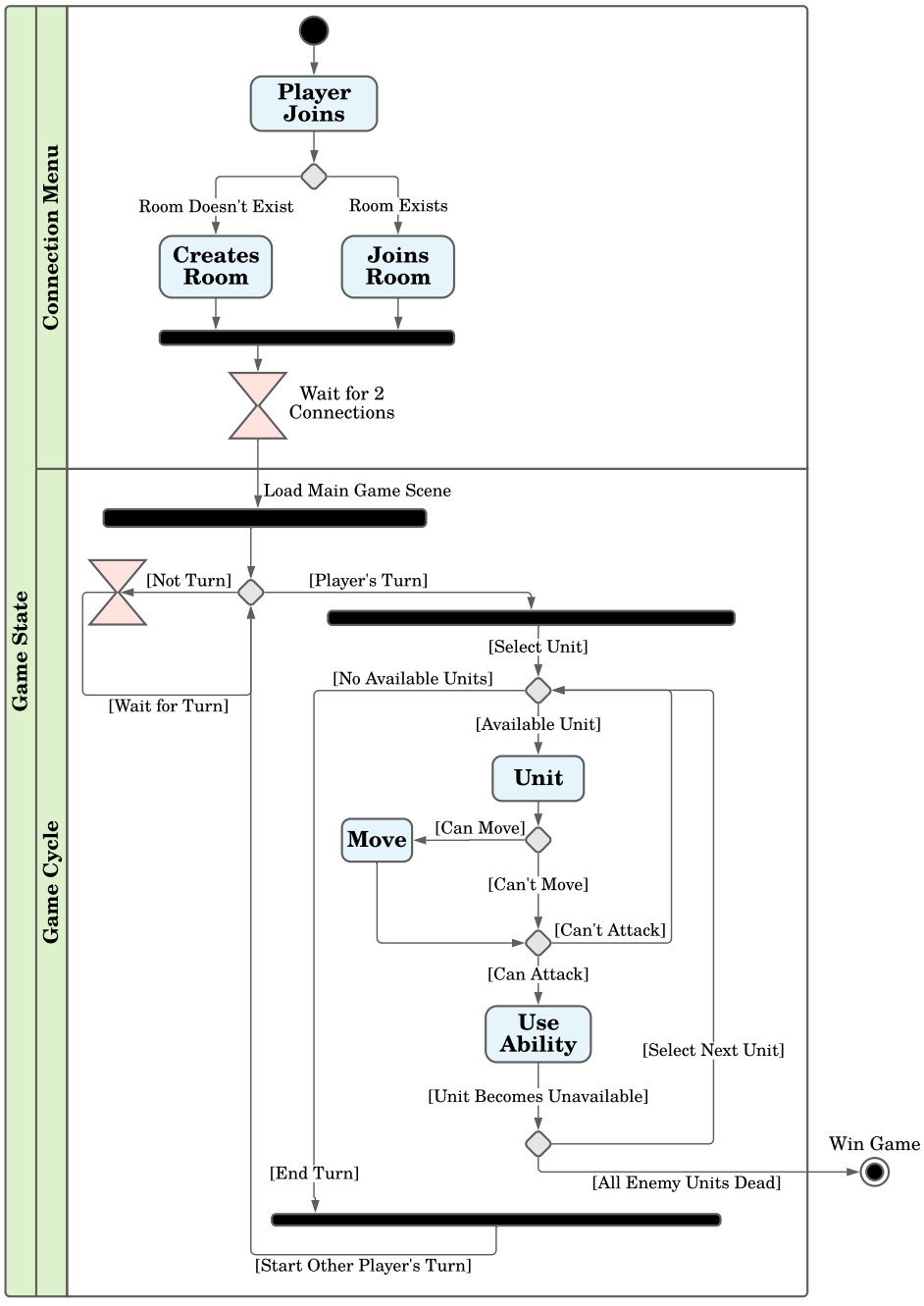


Figure 8: Activity Diagram of a Typical Match

4.6 Visual Design, Contrast and Accessibility

The main UI colour palette was inspired from NieR: Automata – a similarly themed title pivoted around security and machine intelligence. To design an application visually suitable for the web with appropriate complimentary and harmonic colours, this palette was entered into a colour checker [12].

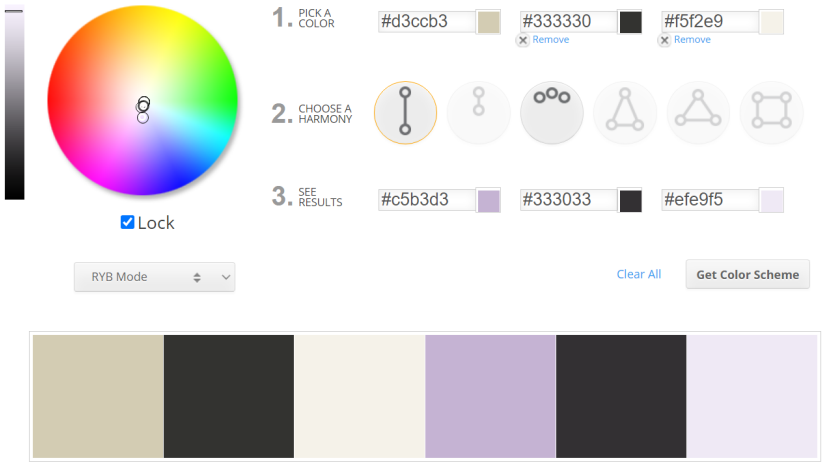
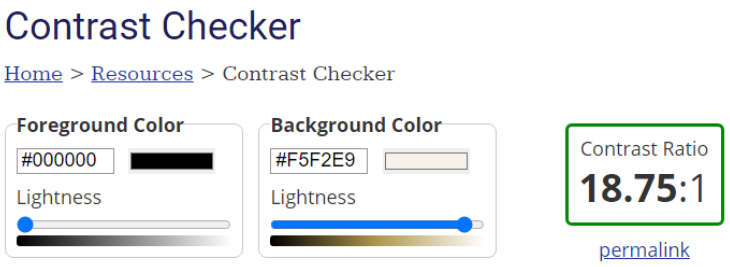
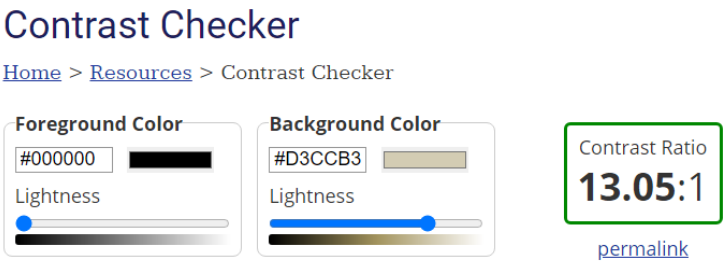


Figure 9: Identifying Complementary and Harmonic Colours

The Web Content Accessibility Guidelines (WCAG 2.1) requires that small text has a minimum contrast ratio of 7:1 [13]. All text used within the application has a contrast ratio of 18.75:1 and 13.05:1 which is compliant for all font-sizes.



(a) Larger Text [14]



(b) Smaller Text [15]

Figure 10: Contract Checking Text for Readability

4.7 Colour Blindness

Approximately 1 in 12 men (8%), and 1 in 200 women (0.5%), experience some form of Colour Vision Deficiency (CVD) [16] which exemplifies the importance of designing with accessibility in mind.

Figure 11 represents the initial design and choice of unit-colour before amending these changes with colour blindness in mind. As such, a colour-blind friendly palette [17], was implemented, alongside a royalty-free background pattern [18], which better contrasts the game-scene.

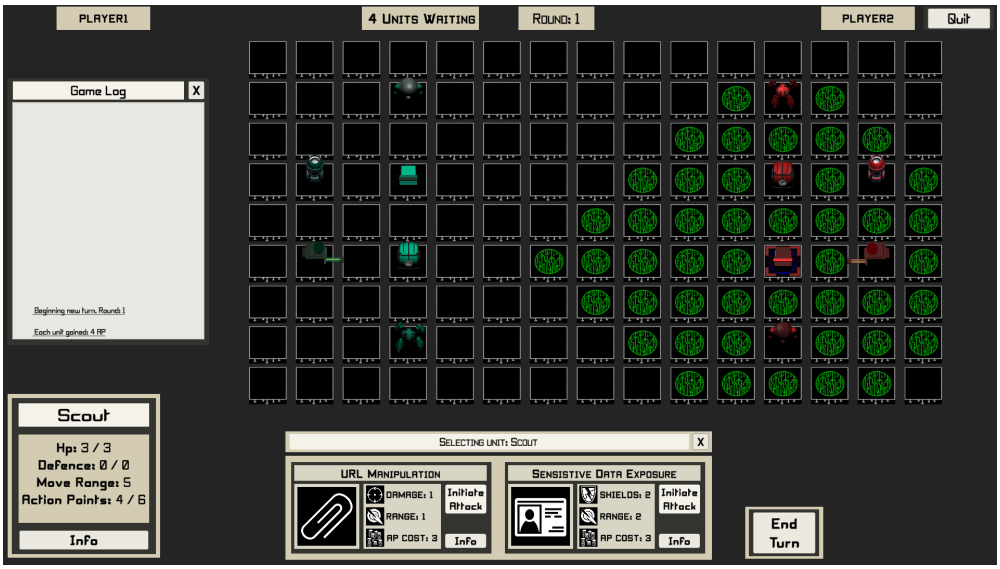


Figure 11: Initial Colour Scheme non-adhering to Accessibility Guidelines

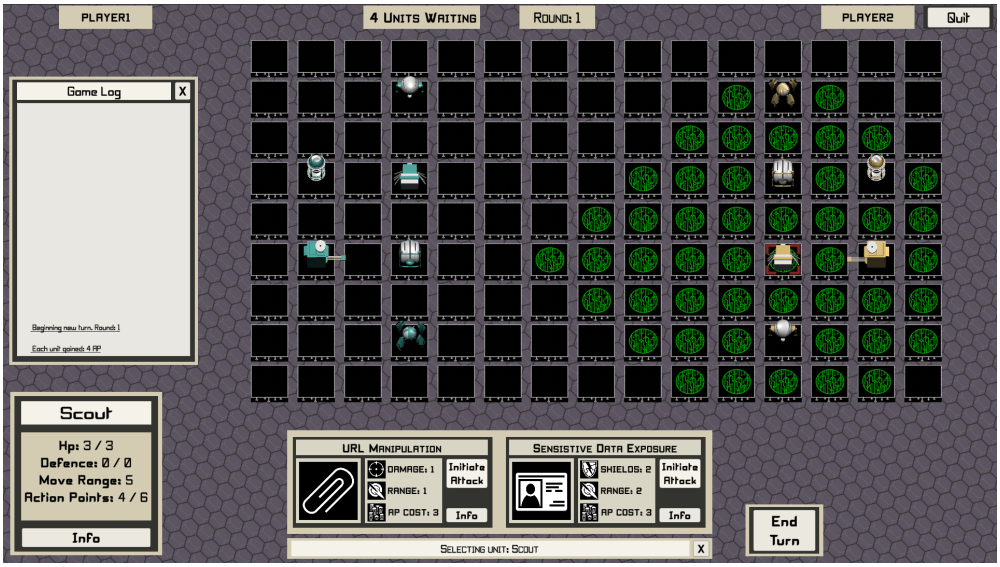


Figure 12: Revised Colour Scheme of the Desktop Application

Due to the technical limitations of WebGL compatibility for web browsers, certain graphical assets (namely the tile design and animation) failed to render within the web browser. As such, an alternative simpler design was developed, as seen in figure 13.

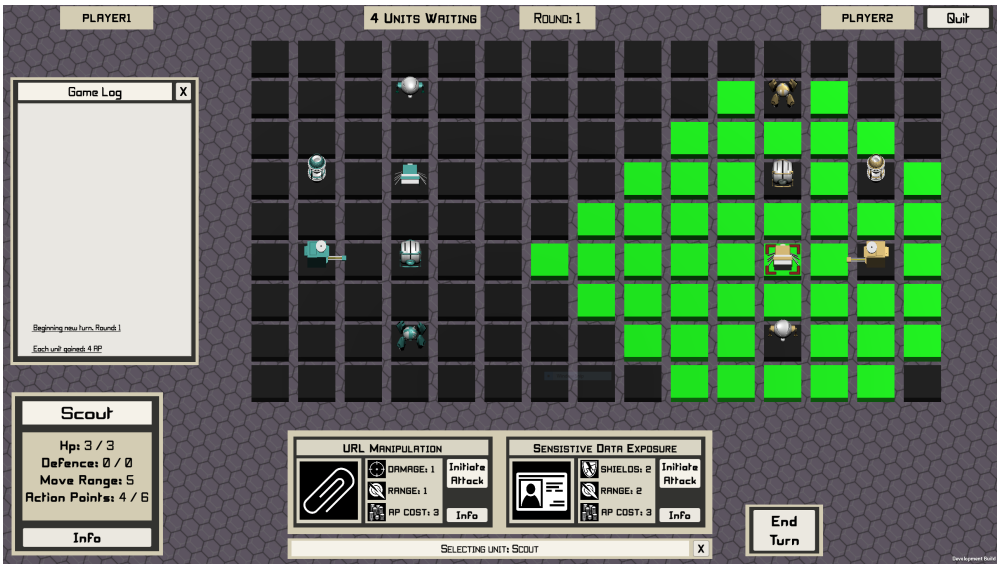


Figure 13: Revised Colour Scheme of the Web Application

4.8 Design Patterns for Unity3D & C#

The following section exemplifies the recommended design patterns for Unity3D.

Should I screenshot sections of my code, or write pseudo-code in a latex environment, which exemplify these patterns?

4.8.1 Singleton Design Pattern

The singleton pattern involves creating a public static reference (of a non-static class) such that only one instance of that class exists [19]. This is necessary for managers (e.g. the Game Manager) as it is not possible to instantiate static classes. However, having a single, static reference is versatile in that it stores important information about the state of the game – and can be publicly invoked by all other components without having to worry about duplications of the class ever occurring.

4.8.2 Flyweight Design Pattern

The flyweight pattern is designed to save memory by re-using instances of the same object [19]. For example, each time a tile changes colour, it is important to update the stored reference of that tile's material. If a material instance is not saved, each time the tile is updated – a new material reference would be created which can lead to a huge accidental memory loss. This is important because as the size of the game scales (with hundreds of tiles), lag spikes will occur more frequently as the WebGL garbage collector must be invoked to remove old material instances.

4.8.3 Component Design Pattern

This is the most common design pattern as Unity is a component-driven engine, in which game objects, scripts and functions are typically broken down into smaller sub-components [19]. The Unit game object is a good example of this as, all unit functionality (stats, movement, abilities, the UI linked to it), are broken down into sub-components and stitched together. Similarly, to object-orientated programming, this approach enables components to be decoupled, re-arranged, and re-used easily.

5 Implementation

Unity3D [20] was chosen as an appropriate development environment because one of the key functional requirements identified was to design a game that could be exported to the web. Unity3D offers seamless integration with WebGL [21] to satisfy this requirement.

Phaser3 [22], HTML and NodeJS were considered as an alternative, web-development based solution. However, Phaser3 is still a relatively small open source project whereas Unity3D is a much more widely used industry trade skill. Furthermore, Unity3D has the largest online community, ample resources and integration for supporting frameworks.

5.1 Client - Server Multiplayer

Photon Cloud is a software as a service (SaaS) solution which provide, and host, cloud servers for free up until 20 concurrent players [23]. Beyond this, Photon offer paid services for more scalable solutions. Furthermore, Photon provides a free-to-use API, (Photon Unity Networking 2 [24]) which enable developers to integrate multiplayer functionality into their games. PUN2 was chosen because it has an ample tutorial base and very strong integration with Unity3D. Lastly, PUN2 supports cross platform gameplay which fulfils the requirement of being able to export and play on the web.

Mirror [25] is an alternative free and open source library which was built to replace Unity's previous deprecated multiplayer libraries (UNET). Although Mirror is completely free to use, relatively simple and highly covered, Mirror does not provide any functionality for externally hosting a game session. This would require developing, and running, a server lobby constantly to allow people to connect to the game. Clearly, this is would not be a viable long-term solution.

5.2 Photon Unity Networking 2 Architecture

PUN2 follows a classic Server – Client architecture. When a client connects to the Photon Network initially, it invokes the 'JoinOrCreateRoom()' function which essentially tries to join a non-full room (if it exists), or create one if there are no other rooms (with space) available.

The first player (client) to do this becomes the master client which is responsible for hosting the game. If the master client disconnects for any reason, there are options for the next available client (if present) to be promoted to a new master client. However, since this project is only designed for two players, it returns the other player to the main menu and ends the room session.

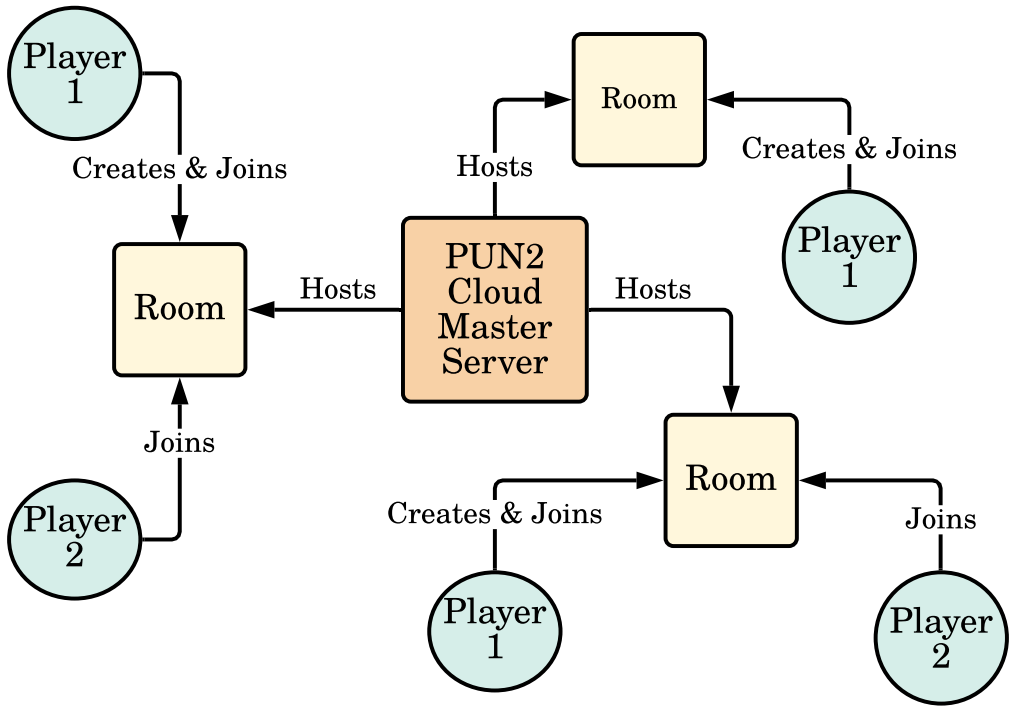


Figure 14: PUN2 Architectural Design

5.3 Remote Procedure Calls - RPCs

PUN RPCs (provided by the PUN2 API [26]) enable client - client communication in which method functions can be invoked on the opponents game version remotely. As both clients are running different, independent images of the game, RPCs are necessary to match these images and important state variables.

For example, after a unit has attacked another unit, it will remotely call the "Take Damage" function which PUN will invoke on the recipient client.

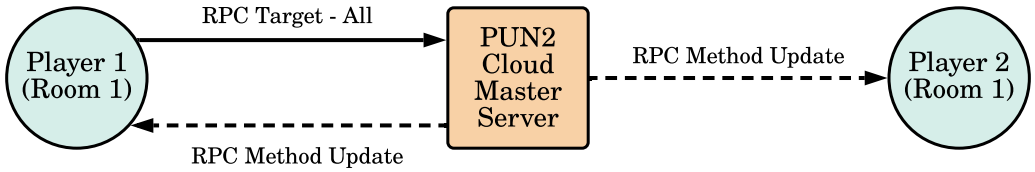


Figure 15: RPC Communication to One User

By calling 'RPC Target.All' however, the function is executed on all clients connected in the room (including the place of origin). This is the most often case since the majority of multiplayer functions are state-changing (such as beginning a new turn, and editing a units states), and these values must remain concurrent.

Figure 16: RPC Communication to All Users

5.4 Movement Mechanics - Breadth First Search

To calculate the selectable tiles in range of a unit, Breadth First Search (BFS) was implemented due to its power to always return the shortest path.

When a unit is selected, a ray cast is sent below to find the current tile that the unit is standing on. From this tile, four more ray casts are emitted to discover any neighbours north, east, south, and west of the current tile; these tiles are added to an adjacency list. Any presently occupied tiles however are ignored as these are not valid tiles from which a unit can move to, or through.BFS is only computed up until the maximum movement range specified by the Unit source.

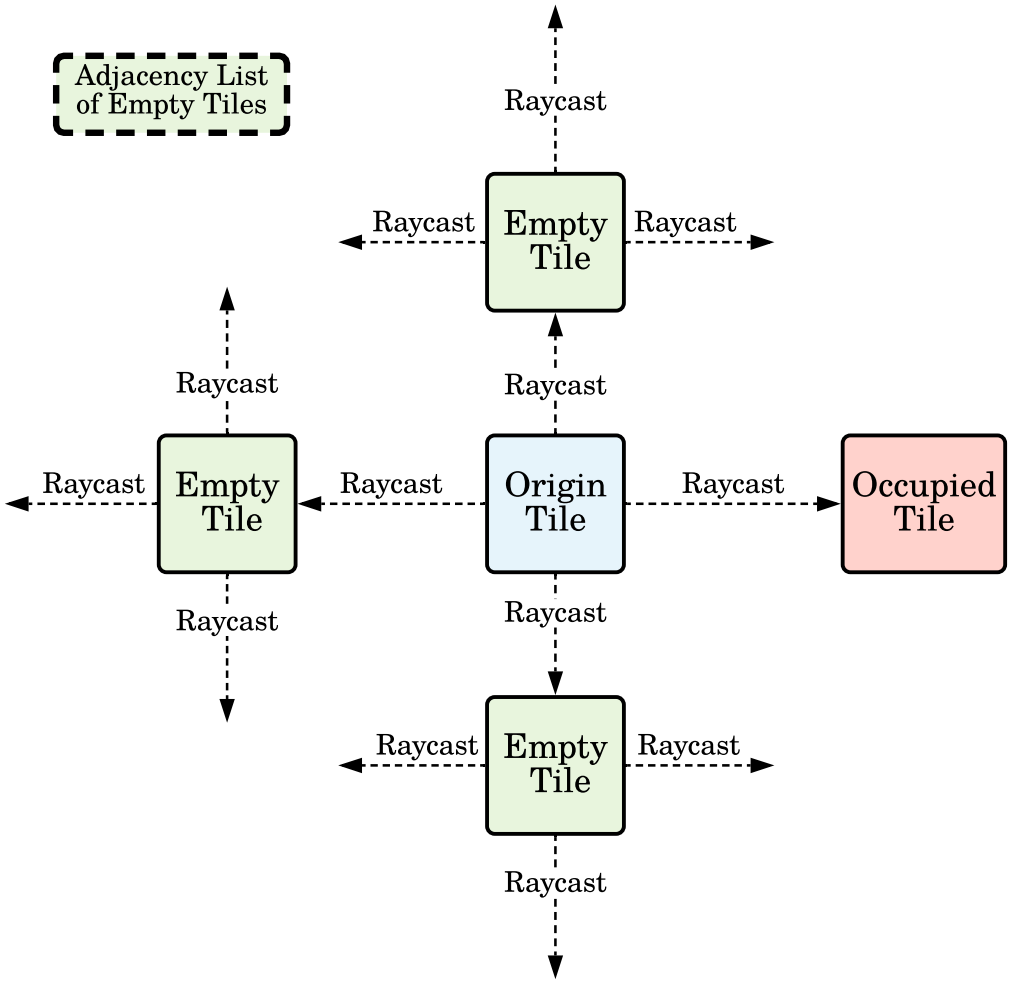


Figure 17: Ray-casting to Adjacent Tiles

BFS in this form utilises a queue data structure in which the current tile is enqueued and checked first, and then every adjacent tile thereafter. For each adjacent tile that is checked, if it is unoccupied, it is flagged as selectable and visited - such that it not checked twice. The diagram below illustrates the operation of checking tiles within a 2 – tile movement range.

If the queue ever reached 0, then the unit would effectively be in a position where it had no legal movies (i.e. it was surrounded by the edge of the map and blocked by other units).

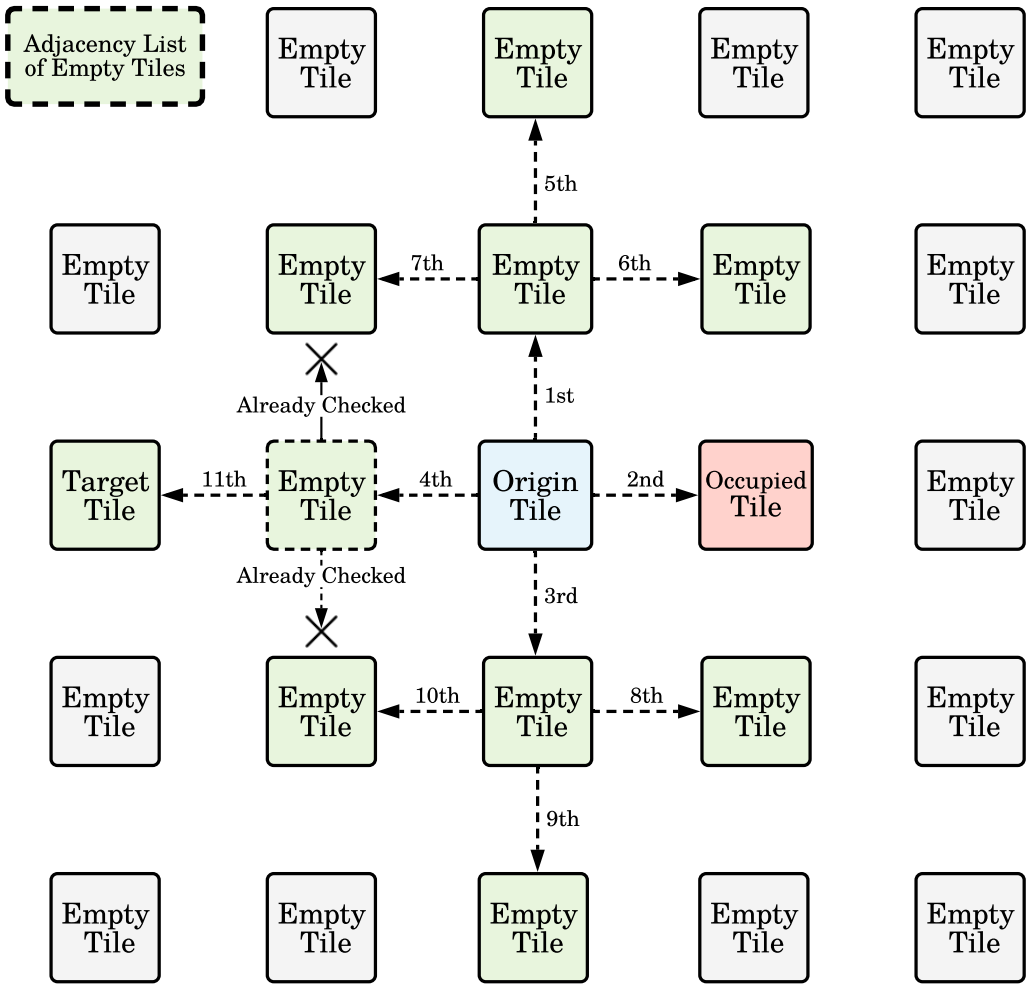


Figure 18: Finding Selectable Tiles via Breadth First Search

For each tile that is dequeued and checked, the parent tile is assigned to its child tile - such that when a user clicks on a selectable tile, we can access its parent recursively until we arrive back at the origin tile. This returns the discovered path to that tile (which is always the shortest path).

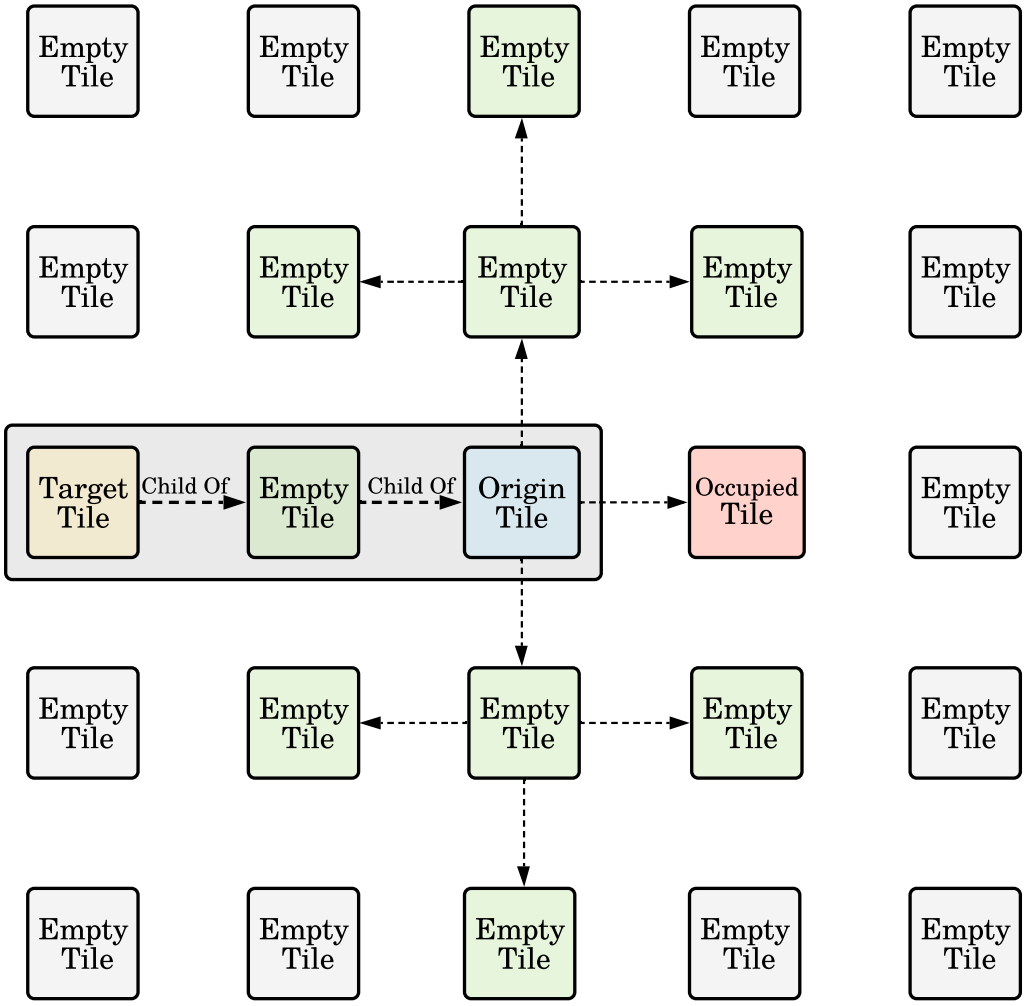


Figure 19: Finding the Shortest Path to Target Tile

Should I include some form of Pseudocode here?, or maybe write it out and add it in the Appendix?

5.5 Implementation of Units

Need to fix the formatting of these sections, and change layout The following sections describe the implementation of selectable units and their following abilities which are derived from the OWASP #10 2017 (cite).

5.5.1 Implementation of Scout Unit

The scout unit is designed to represent exploiting broken access control systems and exposing sensitive information for very poorly designed web applications. As such, it has access to the following moves: URL Manipulation and Sensitive Data Exposure.

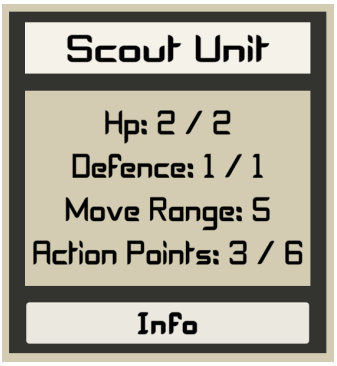


Figure 20: Scout Unit Stats

Sensitive Data Exposure represents OWASP #3. Examples of sensitive data include personal, account and financial information. Improper handling of sensitive data in transit or storage lead to data breaches which can cripple a company's reputation. In more severe outcomes, such information could be used to commit fraud and identity theft. As such, the core ability of this move is to destroy shields of a target unit (at a longer range)

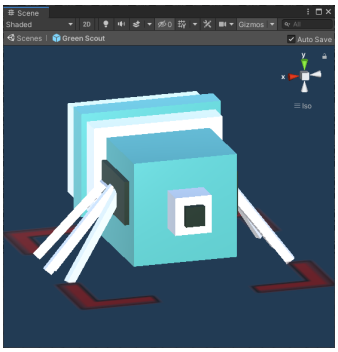


Figure 21: Green Scout

URL Manipulation represents a potential exploit for Broken Access Control - OWASP #5. This attack typically involves changing the URL parameters in order to bypass authorisation control. As such, the core ability of this move is to bypass enemy unit defence, which makes it a good ability to use against other units with high defence, but low HP (such as the back - end database server). This move is close range as it requires the unit to directly interact with the web application - which could potentially be picked up by proper logging and monitoring

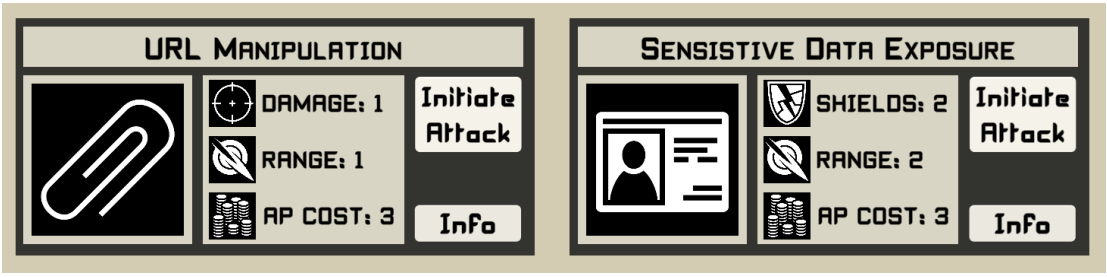


Figure 22: Scout Abilities - URL Manipulation and Sensitive Data Exposure

5.5.2 Implementation of Hacker Unit

The hacker specialises with direct attack vectors typical of penetration testers - that can prod a web application in search of vulnerabilities. SQL injection and XSS scripting are two very common methods that can be brute forced in order to further exploit the web application.

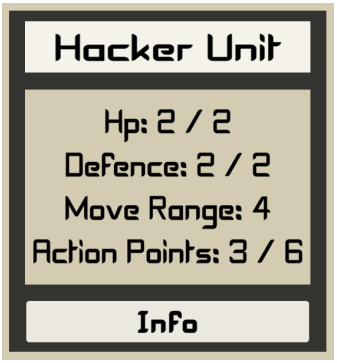


Figure 23: Hacker Stats

As Injection attacks are wanked #1 in the OWASP #10, Sequel Query Language (SQL) injection is likely the most famous, and most common. (SQL) injection exploits the interactivity between the web application and commands that query a backend database. Using SQL injection, it is possible to bypass authorisation, extract sensitive information from databases and even manipulate database entries with your own custom information. With regards to the application, SQL is a slighter longer range physically damaging move.

Cross-Site Scripting (XSS) attacks can occur when a web application fails to validate user input by escaping special characters. As a result, an attacker can execute scripts in a victim's browser, hijack their session data and even redirect them to other phishing sites. With regards to the application, XSS is a closer range physically damaging move.

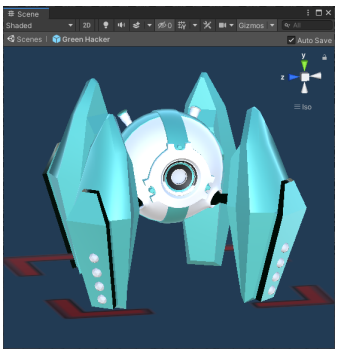


Figure 24: Green Hacker

Figure 25: Hacker Abilities - SQL Injection and XSS Attack

5.5.3 Implementation of Heavy Unit

The Heavy unit is designed to represent the realm of physical disruption by disabling operations externally e.g. denial of service attacks, and internally deleting secu

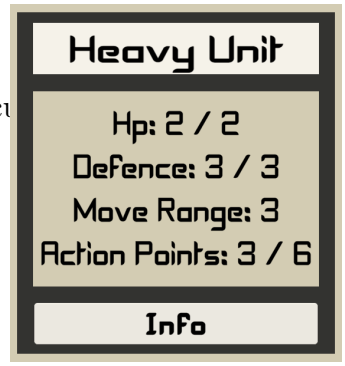


Figure 26: Heavy Unit Stats

A **Distributed Denial of Service** (DDoS) attack involves using a network of devices in order to flood a web application, network or server with overwhelming internet traffic - and disable it from being able to handle requests. **cite**. An attacker can cause denial-of-service by exploiting XML External Entities (XXE) - OWASP #4. As such, this move used on another movable unit will disable it for moving for one turn. If used on the web / database server, action point gain for all units is halted for one turn!.

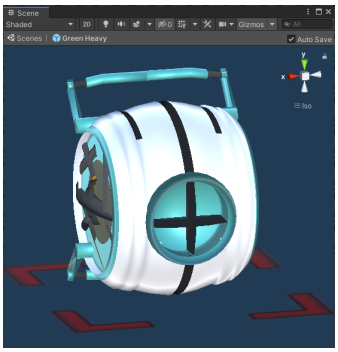


Figure 27: Green Heavy

Delete Security Logs represents further exploiting Insufficient Logging & Monitoring - OWASP # 10. Insufficient data logging and monitoring allows an attacker to persist in a compromised system - undetected for a longer time without being detected. Logging pertains to recording anomalies, failed login attempts, user activity and so on, whilst effective monitoring is to know how to interpret the logged data - and how to react appropriately in the case of such detection. As such, this move is a multi-targeting hitting move (within attack range), that lowers the defence of all of those units.

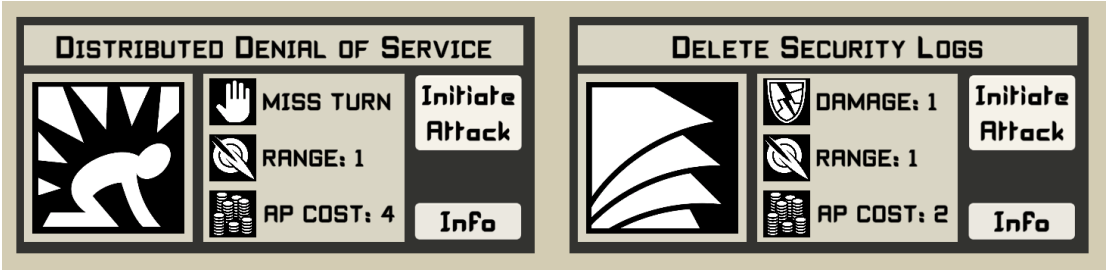


Figure 28: Heavy Abilities - DDoS Attack and Delete Security Logs

5.5.4 Implementation of Analyst Unit

The analyst is designed to represent the role of security analysts who specialise in defending a network. This includes implementing security firewalls and keeping systems up to date. With regards to interactivity with the web application, the analyst is also responsible for monitoring and logging internet traffic in order to detect, and react, to malicious activity.

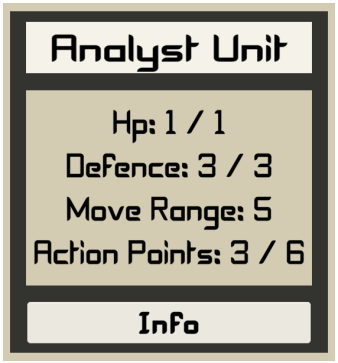


Figure 29: Analyst Stats

Implement Firewall does not directly relate to an attack vector within the OWASP #10, however it was identified as one of the most common solutions to a majority of the attack vectors. A firewall block is required to filter unwanted traffic, and keep web applications components (such as the front-end/back-end) safely in isolation. Furthermore, firewalls can be used in conjunction with an Intrusion Detection/Prevention system in order to detect malicious activity and send alerts upon detection. As such, this move restores the shields of one nearby unit in range.

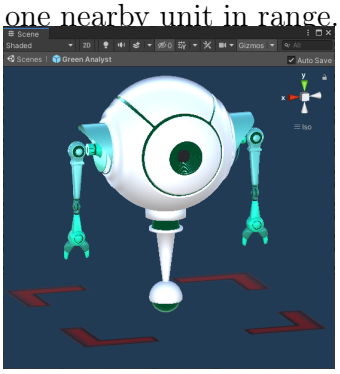


Figure 30: Green Analyst

Implement Access Control is designed as a primary defence mechanism to Broken Access Control - OWASP #5. Broken access control is present when employees/users have access to higher levels of clearance, sensitive information and possibly even "admin" operations than they require. A good access control system should be implemented once, and periodically reviewed. Control policies based on user roles should enforce the least amount of privileges each user requires to complete their interaction within the web application. As such, this move restores the shields of **all** units range.

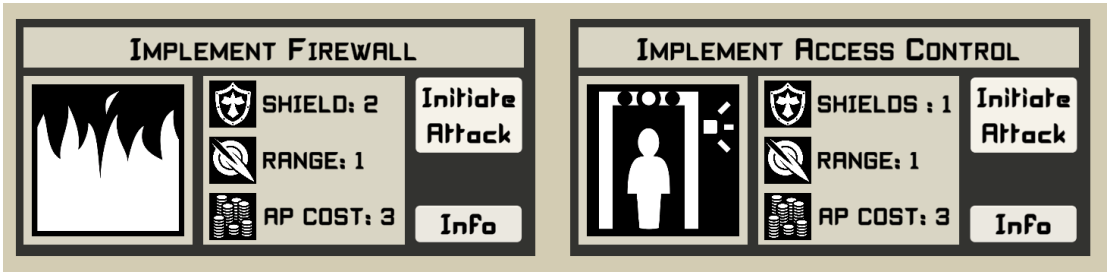
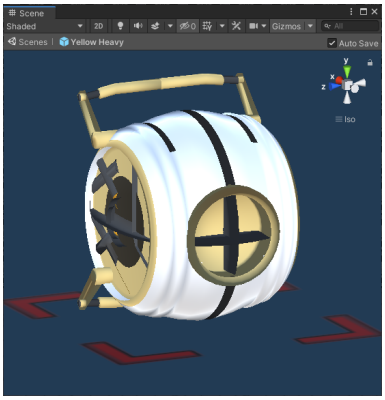


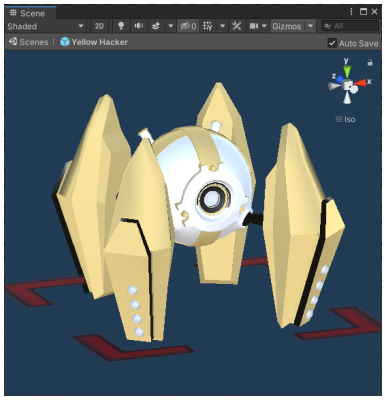
Figure 31: Analyst Abilities - Implement Firewall and Access Control

5.6 Alternative Unit Designs

All unit models were sourced from Blendswap, as royalty-free use models. The colour schemes were changed in line with the colour-friendly design.

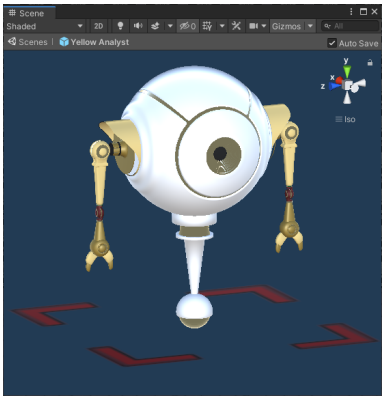


(a) label 1

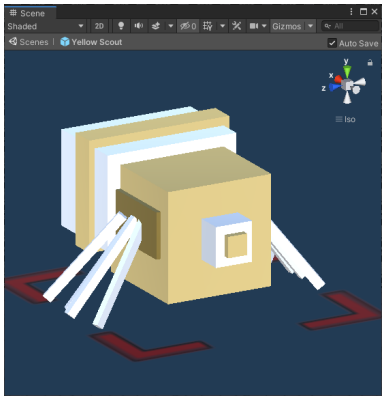


(b) label 2

Figure 32: 2 Figures side by side

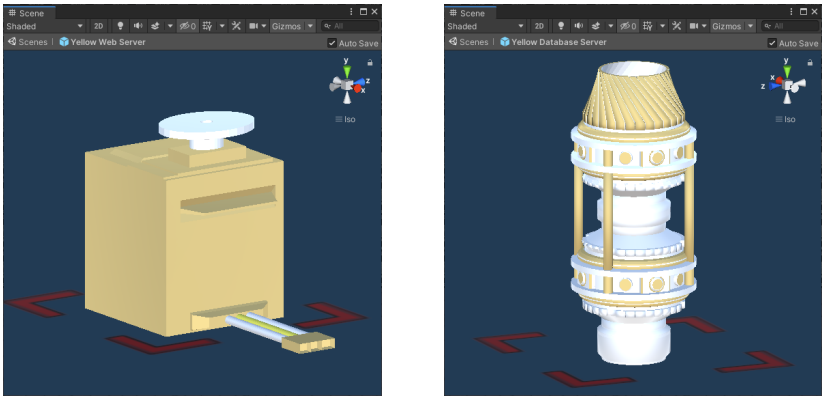


(a) label 1



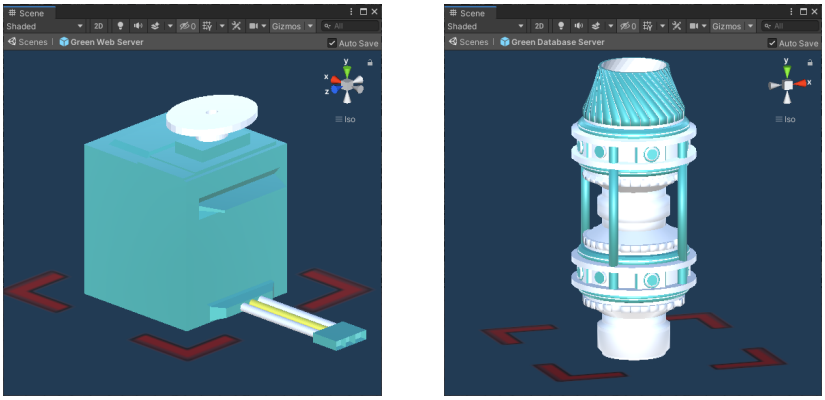
(b) label 2

Figure 33: 2 Figures side by side



(a) label 1 (b) label 2

Figure 34: 2 Figures side by side



(a) label 1 (b) label 2

Figure 35: 2 Figures side by side

6 Testing Strategy

This section explores the testing strategies implemented - which primarily include live debugging (with ParallelSync), unit testing, identifying edges cases and performance analysis (with the Unity profiler).

6.1 Unit Testing

For small single-player games, play-testing in Unity3D is typically very easy via simply running the game. However, as a project expands, it can be a very time-consuming process to launch the game, load menus, connect to another player, and perform actions to reach a desired game state to check one thing. As such, automated testing on pure functions is a necessity.

The following unit tests were performed on all pure functions of the game that should always return the same output on a given input. These include using unique moves with a range of parameters, changing the state of selectable units, and the functional correctness of all calculations (such as calculating damage and action points).

Figure 36: Unity Test Runner

6.2 Identifying Edge Cases

Whilst testing these functions, all edge cases were identified and tested (such as negative, positive and zero values). A large benefit of test-driven development is that it enforces good programming practises in that functions only perform one operation, can be debugged easily and flexibly re-written (without breaking other components). A list of these tests can be seen in Appendix [Ref].

6.3 ParallelSync for Live Builds

ParallelSync [27], is a Unity open source extension which clones the Unity development environment. This is fundamentally important for playtesting a multi-client game as you cannot have duplications of the same environment open concurrently. As a result, this would require rebuilding the project externally which can take several minutes each time (especially for WebGL development builds).

ParallelSync works by cloning the folder structure of the project without needing to duplicate all files, assets, and game scenes. Instead it establishes pointers to the original file location with read only access. Because of this, it is safe and easy to create, load and destroy as many clone environments as required.

6.4 Performance Testing

Performance testing entails utilising the profiling feature which is part of the Unity3D development environment. Performance testing is typically a time consuming process, and as such, is done at the end of a significant feature implementation or project iteration. Profiling is also an iterative and comparative process. At each point, a small "snippet" of the gameplay is captured, this can be saved as a .data file, a change can be made then a sample of this change can be captured in the same way. These cahnges can then be compared to see if the performance has actually dramatically increased or, in fact, decreased. A last point is that it is wise to profile in sustained mode (after the game has run for a while) as weaker devices will downclock the game if it has reached thermal throttling - to a sustainable state.

Insert Section about Structure of a frame here...!

Unity profiling is an instrumentation-based profiler (as opposed to sample-based) [28]. What this means is that a tny marker is inserted in every single function call, such that the time required can be calculated. This creates a tiny bit of overhead, but is otherwise virtually non-existent. The Unity profiler was used to optimise the performance of both the Windows Desktop and WebGL versions of the game, although the WebGL still has some browser specific limitations (more on this will be mentioned later).

Via profiling, the biggest things to look out for any memory loss by the garbage collection, as well as lage spikes.

From profiling, it was discovered that each tile extended from MonoBehaviour and was included in the player loop with an Update Function each turn. What this meant is since my game had 135 tiles, there was a 135 unnecessary tile function calls simply to check the status colour of the tile (and update if necessary) each frame. From this, the code was rearranged such that the tile colours were invoked externally from other functions (such as selecting a unit will highlight all the colours nearby once, and deselecting will reset them).

JetRider itself was chosen as an appropriate IDE for Unity3D, due to it's inbuilt functionality for performance analysis and debugging outside of the Unity development environment. Rider typically offers performance increasing suggestions (by highlighting performance intense tasks) which helped develop in a performance friendly way. With regards to the tile and flightweight pattern mentioned above - caching a tiles material once and accessing this when changing the colour, this is very important for memory andperformance. As when the game scales (and the number of tiles increases to 100s-1000s, if a material is grabbed and changed

without caching it's reference and changing it's reference, a copy is actually made which can quickly lead to memory wastage - especially if called each frame.

The most important optimisation fix however was with regards to how the tiles were discovered. When a unit is selecting, in the UnitController's update loop (called once each frame), it was creating 545 raycasts as it was searching through all adjacent tiles on the map and storing these in memory - each frame! This was causing massive lag spikes in the WebGL version on the game since it was quickly running out of memory and calling garbage collection to remove 21.4kb of memory nearly every frame once the game reached a sustained period (which was almost immediately). This was changed however, such that tiles were cached once at the beginning of the game, and finding selectable tiles via raycasts was only done once each time a unit was selected - leading to almost 0 resources wasted to garbage collection.

6.5 Achieving a Target Frame Rate

In each frame, the core update loop (evaluated by the CPU) will take a certain number of milliseconds and is proportional to the number of operations executed in that frame. Therefore, to achieve a target frame rate, the core update loop must be less than, or equal to, a target budget which is given by:

$$Budget = 1000\ ms\ /\ Target\ Frame\ Rate$$

- To achieve 60 FPS, the target budget would be equal to: $1000\ ms\ /\ 60 = 16\ ms$
- To achieve 30 FPS, the target budget would be equal to: $1000\ ms\ /\ 30 = 32\ ms$

Therefore, the application's target budget is 16ms - such that each frame with the player's update loop is executed in 16ms or less consistently.

Draw a reference diagram of the structure of a frame, illustrating each component

6.6 Validation Testing

In this section compare initial user requirements vs what was completed - and ties into evaluation of project

7 Bibliography

[1] M. Bada and J. R. Nurse, "Developing cybersecurity education and awareness programmes for small-and medium-sized enterprises (smes)," *Information & Computer Security*, 2019.

[2] D. Buil-Gil, F. Miró-Llinares, A. Moneva, S. Kemp, and N. Díaz-Castaño, "Cybercrime and shifts in opportunities during covid-19: a preliminary analysis in the uk," *European Societies*, pp. 1–13, 2020.

[3] H. S. Lallie, L. A. Shepherd, J. R. Nurse, A. Erola, G. Epiphaniou, C. Maple, and X. Bellekens, "Cyber security in the age of covid-19: a timeline and analysis of cyber-crime and cyber-attacks during the pandemic," *arXiv preprint arXiv:2006.11929*, 2020.

[4] J.-N. Tioh, M. Mina, and D. W. Jacobson, "Cyber security training a survey of serious games in cyber security," in *2017 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2017, pp. 1–5.

[5] A. T. Forde, "A gamification toolkit for improving cyber security standards adoption," Master's thesis, University of Southampton, September 2020.

[6] R. Hunicke, M. LeBlanc, and R. Zubek, "Mda: A formal approach to game design and game research," in *Proceedings of the AAAI Workshop on Challenges in Game AI*, vol. 4, no. 1. San Jose, CA, 2004, p. 1722.

[7] C. Koepke, *Persona Image One*. Surface, 2021, Accessed: 19-04-2021. [Online]. Available: https://unsplash.com/photos/E9NcsvbRVqo

[8] Chirstina, *Persona Image Two*. Wocintechat.com, 2021, Accessed: 19-04-2021. [Online]. Available: https://unsplash.com/photos/S3GrMiUhpNU

[9] S. Northcutt, *Persona Image Three*. Unsplash, 2021, Accessed: 19-04-2021. [Online]. Available: https://unsplash.com/photos/sgZX15Da8YE

[10] "Owasp top 10 web application security risks," 2017, Accessed: 20-04-2021. [Online]. Available: https://owasp.org/www-project-top-ten/

[11] G. Miranda, "Unity tutorial - tactics movement," 2017, Accessed: 20-04-2021. [Online]. Available: http://www.gameprogrammingacademy.com/unity-tutorial-tactics-movement/

[12] "Colour Wheel Calculator," 2021, Accessed: 19-04-2021. [Online]. Available: https://www.sessions.edu/color-calculator/

[13] "Web Content Accessibility Guidelines," 2021, Accessed: 19-04-2021. [Online]. Available: https://www.w3.org/TR/2018/REC-WCAG21-20180605/

[14] "Font Contrast Checker," 2021, Accessed: 19-04-2021. [Online]. Available: https://webaim.org/resources/contrastchecker/?fcolor=000000&bcolor=D3CCB3

[15] "Font Contrast Checker," 2021, Accessed: 19-04-2021. [Online]. Available: https://webaim.org/resources/contrastchecker/?fcolor=000000&bcolor=F5F2E9

[16] "Colour blindness awareness," 2021, Accessed: 19-04-2021. [Online]. Available: https://www.colourblindawareness.org/

[17] D. Nichols, "Coloring for colorblindness," 2021, Accessed: 19-04-2021. [Online]. Available: https://davidmathlogic.com/colorblind/#%23E1BE6A-%2340B0A6

[18] "Doodad Pattern Generator," 2021, Accessed: 19-04-2021. [Online]. Available: https://doodad.dev/pattern-generator/

[19] D. Baron, *Hands-On Game Development Patterns with Unity 2019: Create engaging games by using industry-standard design patterns with C.* Packt Publishing Ltd, 2019.

[20] "Unity 3d," 2021, Accessed: 19-04-2021. [Online]. Available: https://unity.com/

[21] "Unity - manual: Getting started with webgl development," 2021, Accessed: 19-04-2021. [Online]. Available: https://docs.unity3d.com/Manual/webgl-gettingstarted.html

[22] "Phaser3," 2021, Accessed: 19-04-2021. [Online]. Available: https://phaser.io/phaser3

[23] "Photon cloud vs photon server," 2021, Accessed: 19-04-2021. [Online]. Available: https://doc.photonengine.com/en-us/realtime/current/getting-started/onpremises-or-saas

[24] "Pun2 documentation," 2021, Accessed: 19-04-2021. [Online]. Available: https://doc-api.photonengine.com/en/PUN/v2/index.html

[25] "Mirror networking," 2021, Accessed: 19-04-2021. [Online]. Available: https://mirror-networking.gitbook.io/docs/

[26] "Rpcs and raiseevents," 2021, Accessed: 19-04-2021. [Online]. Available: https://doc.photonengine.com/en-us/pun/v2/gameplay/rpcsandraiseevent

[27] "Parralelsync - github," 2021, Accessed: 19-04-2021. [Online]. Available: https://github.com/VeriorPies/ParrelSync

[28] "Unity - manual: The profiler window," 2021, Accessed: 19-04-2021. [Online]. Available: https://docs.unity3d.com/Manual/ProfilerWindow.html